What is claimed is:

1. A processor comprising:

a plurality of execution units to allow execution of a plurality of threads, including a first thread, said first thread having a first instruction having an associated address operand indicating a monitor address;

suspend logic to suspend execution of said first thread;

a monitor to cause resumption of the first thread in response to a memory access to the monitor address.

2. The processor of claim 1 wherein said monitor is to cause resumption in response to the memory access only if the memory access indicates an actual or potential write to the monitor address.

3. The processor of claim 1 wherein the monitor is to cause resumption of the first thread in response to the memory access to the monitor address if the first thread is suspended and monitor events are unmasked.

4. The processor of claim 3 further comprising event detection logic to cause resumption of said first thread in response to an event other than said memory access.

5. The processor of claim 4 wherein said event is an interrupt.

6. The processor of claim 1 wherein said associated address operand is an implicit

operand.

7. The processor of claim 6 wherein said associated address operand is stored in a predetermined register.

8. The processor of claim 1 wherein said suspend logic is to suspend execution of said first thread in response to a second instruction, wherein the first instruction enables the monitor and the second instruction unmasks events signaled by the monitor.

9 The processor of claim 8 wherein said second instruction only enables the monitor if the first instruction has been executed.

10. The processor of claim 1 wherein said suspend logic is to suspend execution of said first thread in response to the first instruction.

11. The processor of claim 8 further comprising:

coherency logic to improve visibility of stores to said monitor address.

12. The processor of claim 11 wherein said coherency logic is to ensure that no cache within a coherency domain stores information at said monitored address in a modified or exclusive state.

13. The processor of claim 12 wherein said coherency logic is to flush a cache line

associated with said monitored address from any internal caches and to generate a bus read line transaction of the cache line associated with said monitor address to other processors coupled to the processor, the bus read line transaction being a multi-phase transaction provided according to a pipelined bus protocol.

14. The processor of claim 11 wherein said coherency logic is to cause said processor to generate a bus cycle to prevent any other bus agents from performing a write transaction to said monitor address without broadcasting the write transaction.

15. The processor of claim 14 further comprising bus control logic to assert a hit signal in response to another bus agent reading information at said monitor address.

16. The processor of claim 1 wherein said monitor address indicated by said associated address operand indicates one of a cache line, a portion of a cache line, or an alternatively sized unit, for data at an address indicated by said associated address operand.

17. The processor of claim 1 further comprising address translation logic to translate said associated address operand to the monitor address which is a physical address.

18. The processor of claim 1 wherein said monitor address is chosen from a set consisting of a physical address, a virtual address, a relative address, and a linear address.

19. The processor of claim 1 further comprising a plurality of partitionable resources to be partitioned to dedicate a portion of each partitionable resource to each active one of said plurality of threads when multiple threads are active, wherein said suspend logic is to relinquish any of said plurality of partitions dedicated to said first thread in response to suspending execution of said first thread.

20. The processor of claim 19 wherein the monitor is to cause said plurality of partitionable resources to be re-partitioned to accommodate execution of said first thread in response to the memory access to the monitor address.

21. The processor of claim 20 wherein said plurality of partitionable resources comprise:

an instruction queue;

a re-order buffer;

a pool of registers;

a plurality of store buffers.

22. The processor of claim 21 further comprising:

a plurality of duplicated resources, said plurality of duplicated resources being duplicated for each of said plurality of threads, said plurality of duplicated resources comprising:

a plurality of processor state variables;

an instruction pointer;

register renaming logic.

23. The processor of claim 22 further comprising:

a plurality of shared resources, said plurality of shared resources being available for use by any of said plurality of threads, said plurality of shared resources comprising:

said plurality of execution units;

a cache;

a scheduler.

24. A processor comprising:

a front end to receive a first instruction from a first thread and a second instruction from the first thread, the first instruction indicating a monitor address;

execution resources to execute the first instruction and the second instruction and to suspend execution of said first thread in response to the second instruction;

a monitor to cause resumption of the first thread in response to a memory access to the monitor address.

25. The processor of claim 24 wherein said first instruction has an operand indicating a linear address, and wherein said processor further comprises address translation logic to translate said linear address to obtain the monitor address which is a physical address.

26. The processor of claim 25 further comprising:

coherency logic to ensure that no cache in another processor coupled to the processor stores information at said monitor address in a modified or exclusive state.

27. The processor of claim 26 wherein said coherency logic is to assert a hit signal in response to another processor snooping the monitor address.

28. A processor comprising:

front end logic to receive a first instruction from a first thread, the first instruction having an associated monitor address;

a monitor coupled to receive the monitor address, and in response to said first instruction to monitor memory accesses to said monitor address and to signal an event when an access to said monitor address occurs.

29. The processor of claim 28 wherein said monitor is to signal the event in response to a write memory access that writes to the monitor address.

30. The processor of claim 28 wherein said monitor is to signal the event in response to a line invalidating transaction.

31. The processor of claim 28 further comprising:

coherency logic to ensure that no cache in another processor coupled to the

processor stores information at said monitor address in a modified or exclusive state.

32. The processor of claim 31 wherein said coherency logic comprises logic to generate an internal cache flush cycle and to generate an external read line transaction.

33. The processor of claim 28 further comprising:

logic to unmask monitor events from said monitor and to suspend said first thread in response to a second instruction.

34. A processor comprising:

a plurality of execution units to execute a plurality of threads;

front end logic to receive an instruction from a first thread of said plurality of threads;

suspend logic to suspend said first thread in response to said instruction if no monitor events are pending, and to allow other ones of said plurality of threads to execute.

35. The processor of claim 34 wherein said suspend logic is to enable recognition of monitor events, including already pending monitor events.

36. The processor of claim 35 wherein said processor comprises a plurality of partitionable resources, and wherein said suspend logic is to relinquish partitions of

each of said plurality of partitionable resources associated with said first thread in addition to suspending the first thread in response to the instruction.

37. A processor comprising:

a plurality of thread partitionable resources to receive instructions;

a plurality of shared resources to execute instructions in cooperation with said plurality of thread partitionable resources;

thread suspension logic to suspend a first thread in response to an instruction in said first thread, said thread suspension logic to relinquish partitions of said plurality of thread partitionable resources associated with the first thread in addition to suspending the first thread;

a monitor to cause said processor to re-partition said plurality of thread partitionable resources and to resume execution of said first thread in response to an access to a memory address indicated by the first thread.

38. The processor of claim 37 wherein said access to the memory address is specified by a first instruction executed in said first thread and wherein the monitor is unmasked to signal monitor events to cause thread resumption by the instruction in response to which the thread suspension logic is to suspend the first thread.

39. An apparatus comprising:

means for suspending a first thread of a plurality of threads of execution;

means for detecting an access to a memory location;

means for resuming said first thread in response to the means for detecting detecting the access to the memory location.

40. The apparatus of claim 39 wherein said means for detecting the access to the memory location is enabled in response to a first instruction executed in said first thread, and wherein said means for suspending the first thread suspends the first thread in response to a second instruction executed in said first thread.

41. The apparatus of claim 40 further comprising:

coherency means for simplifying detection of the access to the memory location.

42. The apparatus of claim 41 wherein said access to the memory location is a write or an invalidating access.

43. The apparatus of claim 41 further comprising:

means for annealing resources in response to the means for suspending suspending execution of said first thread, said means for annealing freeing partitioned resources associated with said first thread for use by other ones of said plurality of threads;

means for partitioning resources for re-partitioning resources to accommodate resumption of said first thread.

44. A method comprising:

receiving a first opcode in a first thread of execution, said first opcode having an associated address operand which indicates a monitor address;

suspending said first thread;

detecting a memory access to the monitor address;

resuming said first thread in response to detecting the memory access to the monitor address.

45. The method of claim 44 wherein suspending the first thread comprises:

receiving a second instruction in the first thread;

suspending the first thread in response to the second instruction.

46. The method of claim 45 wherein the memory access is a write access.

47. The method of claim 45 further comprising translating said associated address operand into a monitored physical address, wherein detecting the memory access to the monitor address comprises detecting a write access to the monitored physical address.

48. The method of claim 44 further comprising:

preventing other agents from obtaining ownership of information stored at said monitor address.

49. The method of claim 44 wherein detecting comprises:

receiving cycle information from external bus transactions;

detecting writes to the monitor address.

50. The method of claim 44 further comprising:

resuming said first thread in response to an event other than the memory access to the monitor address.

51. The method of claim 50 wherein said event is an interrupt.

52. The method of claim 51 wherein said interrupt is a masked interrupt that is indicated by a second operand to nonetheless be considered a break event.

53. A method comprising:

receiving a first opcode executing in a first thread of execution;

translating a linear address associated with said first opcode into a physical address;

executing a bus transaction by a monitoring bus agent to ensure no other bus agent has sufficient ownership of data associated with said physical address to allow another bus agent to modify the data without informing the monitoring bus agent;

monitoring for a write access to said physical address;

signaling a hit if another bus agent reads said physical address;

receiving a second opcode in the first thread of execution;

suspending said first thread of execution and enabling recognition of a monitor event in response to the second opcode;

resuming said first thread if the write access occurs;

resuming execution of the first thread in response to any one of a first set of events;

ignoring a second set of events.

54. The method of claim 53 wherein suspending the first thread of execution in response to the second opcode comprises:

testing whether the monitor event is pending;

testing whether a monitor is active;

if the monitor is active and no monitor event is pending, then entering a first thread suspended state.

55. The method of claim 54 wherein entering the first thread suspended state comprises:

relinquishing a plurality of registers in a register pool;

relinquishing a plurality of instruction queue entries in an instruction queue;

relinquishing a plurality of store buffer entries in a store buffer;

relinquishing a plurality of re-order buffer entries in a re-order buffer.

56. A system comprising:

a memory to store a first instruction from a first thread, the first instruction having an associated address operand indicating a monitor address;

a first processor coupled to said memory, said first processor to enable a monitor to monitor memory transactions to detect a memory access to said monitor address in response to the first instruction and to cause resumption of said first thread in response to the memory access to the monitor address.

57. The system of claim 56 wherein said memory is to store a second instruction from said first thread, and wherein said first processor is to suspend said first thread in response to the second instruction.

58. The system of claim 57 wherein said monitor is to set a monitor event pending indicator in response the memory access occurring, said monitor event pending indicator to cause said first processor to resume a thread once unmasked by said second instruction.

59. The system of claim 56 wherein said first processor includes a first cache, the system further comprising:

a second processor comprising a second cache, wherein said first processor drives a bus transaction to the second processor to force said second processor to broadcast to the first processor any transactions that allow alteration of data stored at the monitor address in the second cache.

60. The system of claim 59 wherein said first processor is to assert a signal preventing said second processor from caching data at the monitor address in a state which would allow the second processor to modify data stored at the monitor address in the second cache without broadcasting that a modification is occurring.

61. The system of claim 60 wherein said signal indicates a cache hit and prevents the second cache from storing data at the monitor address in an exclusive state.

62. The system of claim 58 wherein said first processor is further to resume the first thread if an alternative event occurs.

63. The system of claim 62 wherein said alternative event is an interrupt.

64. The system of claim 62 wherein said first thread stored in said memory includes a loop, said loop including the first instruction and the second instruction as well as a test to determine if data at the monitor address has changed and to re-start the loop if data at the monitor address remains unchanged.

65. An article comprising a computer readable medium, which represents a processor comprising:

a plurality of execution units to allow execution of a plurality of threads, including a first thread, said first thread having a first instruction having an associated address operand indicating a monitor address;

suspend logic to suspend execution of said first thread;

a monitor to cause resumption of the first thread in response to a memory access to the monitor address.

66. The article of claim 65 wherein said monitor is to cause resumption in response to the memory access only if the memory access indicates an actual or potential write to the monitor address.

67. The article of claim 65 wherein the monitor is to cause resumption of the first thread in response to the memory access to the monitor address if the first thread is suspended and monitor events are unmasked.

68. The article of claim 65 wherein said processor further comprises event detection logic to cause resumption of said first thread in response to an event other than said memory access.

69. The article of claim 68 wherein said processor further comprises a plurality of partitionable resources to be partitioned to dedicate a portion of each partitionable resource to each active one of said plurality of threads when multiple threads are active, wherein said suspend logic is to relinquish any of said plurality of partitions dedicated to said first thread in response to suspending execution of said first thread.